UNIVERSIDAD PERUANA DE CIENCIAS APLICADAS

SECCIÓN: CC52

GRUPO: 3

CURSO: Fundamentos de Data Science

PROFESOR(A): Nérida Isabel Manrique Tunque

TÍTULO: Trabajo Parcial Fundamentos de Data Science

El presente trabajo ha sido realizado por:

Joaquín Eduardo Velarde Leyva U202212510

Victor Daniel Chipana Gutierrez U202115805

Daniel Ivan Carbajal Robles U20221B751

Ian Joaquin Sanchez Alva U202124676

2024-01

Índice:

I. Introducción

El análisis exploratorio de datos (EDA) es una etapa crucial en cualquier proyecto de análisis de datos. En este informe, se llevará a cabo un EDA utilizando RStudio como herramienta de software sobre el conjunto de datos "Hotel booking demand".

El objetivo de este análisis exploratorio de datos es comprender en profundidad el conjunto de datos "Hotel booking demand" a través de técnicas de visualización, preparación y análisis en RStudio. Se busca identificar patrones y tendencias en las reservas de hotel, evaluar la calidad de los datos abordando valores faltantes y datos atípicos, explorar relaciones entre variables como la duración de la estadía y la disponibilidad de estacionamiento, comparar el comportamiento entre el hotel urbano y el resort, y obtener insights básicos para la toma de decisiones estratégicas en la gestión hotelera.

II. Caso de análisis

El origen de los datos se encuentra en diversas fuentes, como las reservas de habitaciones, las reseñas de huéspedes, los datos de ventas y las estadísticas de ocupación. Los datos se recolectan a través de diferentes herramientas y sistemas, como las reservas centrales, los sistemas de gestión de reservas y los sistemas de gestión de datos. La recopilación de datos fue realizada por Nuno Antonio, Ana de Almeida y Luis Nunes

Diccionario de variables:

Nombre	Tipo	Descripción
hotel	character	Nombre de los hoteles (Resort Hotel, City Hotel)

is_canceled	integer	Indica si se cancela la reserva (0 no se canceló, 1 se canceló)
lead_time	integer	Número de días transcurridos entre la fecha de reserva con la fecha de llegada
arrival_date_year	integer	Año de fecha de llegada
arrival_date_month	character	Mes de fecha llegada
arrival_date_week_number	integer	Número de semana de la fecha llegada
arrival_date_day_of_month	integer	Día del mes de la fecha de llegada
stays_in_weekend_nights	integer	Número de noches que el huespede se quedó o reservó en el hotel para fin de semana
stays_in_week_nights	integer	Número de noches que el huespede se quedó o reservó en el hotel entre semana
adults	integer	Número de adultos
children	integer	Número de niños
babies	integer	Número de bebes
meal	character	Tipo de comida reservada (BB, FB, HB, SC, Undefined)
country	character	País de origen
market_segment	character	Designación del segmento de mercado
distribution_channel	character	Canal de distribución de cocina
is_repeated_guest	integer	Valor que indica si el nombre de la reserva fue de alguien repetido (1 -> si, 0 -> no)

previous_cancellations	integer	Número de reservas anteriores que fueron cancelados por el cliente antes de la reserva actual
previous_bookingsd_not_canceled	integer	Número de reservas anteriores no cancelados por el cliente antes de la reserva actual
reserved_room_type	character	Código del tipo de habitación de reserva
assigned_room_type	character	Código del tipo de habitación asignado al reservar
booking_changes	integer	Número de cambios realizadas a la reserva desde el momento en que la reserva se realizó hasta el momento de cancelación o entrega
deposit_type	character	Indicación si el cliente realizó un depósito para garantizar la reserva
agent	character	ID de la agencia de viajes que realizó la reserva
company	character	ID de la compañía que realizó la reserva o responsable del pago de la reserva
days_in_waiting_list	integer	Número de días que la reserva estuvo en la lista de espera antes de confirmarse por el cliente
customer_type	character	Tipo de reserva(contract, group, transient, transient-party)
adr	numeric	Tarifa diaria promedio
required_car_parking_spaces	integer	Número de plazas de aparcamiento necesarias para el cliente

total_of_special_request	integer	Número de solicitudes especiales realizadas por el cliente
reservation_status	character	Último estado de la reserva (Canceled, Check-Out, No-Show)
reservantion_status_date	date	Fecha en la que se cambió el último estado

La herramienta utilizada fue RStudio, software que usa el lenguaje para análisis de datos “R”. Gracias a las funciones de las librerías, que se pueden descargar en el software, tiende a facilitar el análisis de los datasets. Las librerías te ayudan a facilitar diversas tareas. Por ejemplo, ggplot2 es una librería utilizada para graficar, y otras más.

Los casos de uso aplicable son:

- Predicción de Cancelaciones: El caso principal es predecir la probabilidad de que una reserva de hotel sea cancelada.
- Pronóstico de Demanda: Utilizando los datos históricos de reservas, se puede pronosticar la demanda de habitaciones de hotel para fechas futuras, asimismo, el tipo de reserva que se hace como, cantidad de adultos, niños, bebes, si pagó un adelanto, etc.
- Análisis de Mercado: Al comparar el rendimiento del hotel con el de la competencia y analizar las tendencias del mercado.

Desarrollo de preguntas de análisis:
Primero para responder estas preguntas se hizo lo siguiente:

Lectura de los datos desde la ruta (Los espacios vacíos fueron reemplazos por NA):

```
setwd("C:/Users/Usuario/Desktop/Querys de R")
hotels<- read.csv('hotel_bookings.csv', header = TRUE , sep = ',',
dec = '.',stringsAsFactors = FALSE , na.strings = "")
```

Limpieza de datos (Se utiliza ese comando para poder omitir los NA y tener una data limpia):

```
hotels_data.limpia <- na.omit(hotels)
```

Y se usaron estas librerías:

```
install.packages("ggplot2")
install.packages("dplyr")
library(ggplot2)
library(dplyr)
```

A. ¿Cuántas reservas se realizan por tipo de hotel? o ¿Qué tipo de hotel prefiere la gente?

```
##Este data frame solo tiene los hoteles y las reservas
datos <- data.frame(
  hotels_data.limpia$hotel ,
  hotels_data.limpia$is_canceled
)
```

head(datos)

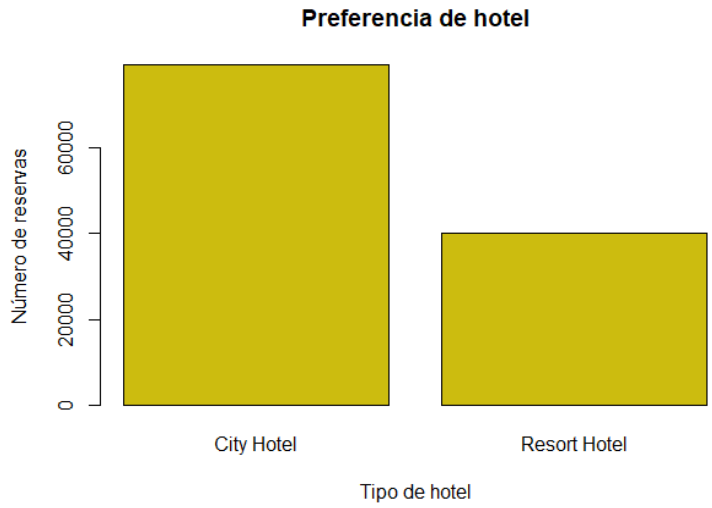
```
 hotels_data.limpia.hotel hotels_data.limpia.is_canceled
1             Resort Hotel                              0
2             Resort Hotel                              0
3             Resort Hotel                              0
4             Resort Hotel                              0
5             Resort Hotel                              0
6             Resort Hotel                              0
```

```
conteo_por_hotel <- table(datos$hotels_data.limpia.hotel)
conteo_por_hotel
City Hotel Resort Hotel
       79330        40060
```

Con esta tabla “conteo_por_hotel” podemos responder a la pregunta de cuántas reservas se realizan por hotel. Y concluimos que en el tipo de hotel “City Hotel” se realizaron 79330 reservas y en “Resort Hotel” 40060.

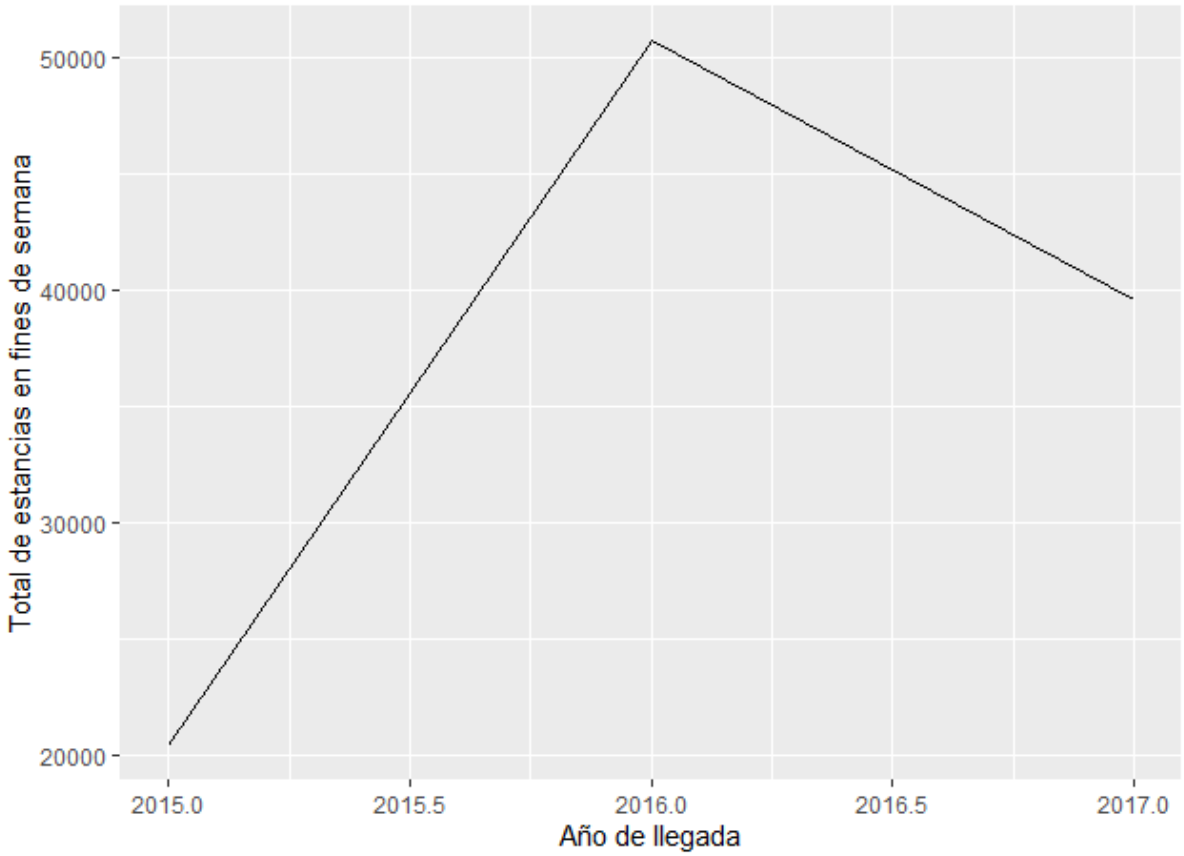
```
barplot(conteo_por_hotel,
        main = "Preferencia de hotel",
        xlab = "Tipo de hotel",
        ylab = "Número de reservas",
        col = "#CCBC0F",
        border = "black")
```

Con este gráfico se puede observar la diferencia y también concluir que la gente prefiere el tipo “City Hotel”



B. ¿Está aumentando la demanda con el tiempo?
Esta línea de código nos permite ordenar los datos por año de mayor a menor

```
datos_agrupados <- group_by(hotels_data.limpia,
arrival_date_year)

##calcular la suma de cuantas veces se quedaron en fines de
semana por año


demanda_por_anio <- summarise(datos_agrupados,
total_stays_in_weekend_nights = sum(stays_in_weekend_nights))

arrival_date_year total_stays_in_weekend_nights
              <int>                        <int>
1              2015                        20450
2              2016                        50695
3              2017                        39601

ggplot(demanda_por_anio, aes(x = arrival_date_year, y =
total_stays_in_weekend_nights)) +
  geom_line() +
  labs(x = "Año de llegada", y = "Total de estancias en fines
de semana")
```



Con el gráfico y la tabla de demanda_por_anio podemos concluir que la tendencia baja después del año 2016.

C. ¿Cuándo se producen las temporadas de reservas: alta, media y baja?

```
## convertir los meses a factor para que puedan ser ordenados
hotels_data.limpia$arrival_date_month <-
factor(hotels_data.limpia$arrival_date_month, levels =
c("January", "February", "March", "April", "May", "June",
"July", "August", "September", "October", "November",
"December"))

##Se hace un conteo  de las reservas por mes
reservas_por_mes <- table
(hotels_data.limpia$arrival_date_month)
```

```
 January  February     March     April       May      June      July    August September   October  November  December
    5929      8068      9794     11089     11791     10939     12661     13877     10508     11160      6794      6780
```

```
## sacamos el promedio de la tabla anterior para cada mes
promedio_reservas <- mean(reservas_por_mes)
promedio_reservas
[1] 9949.167

## le damos valores para definir las temporadas
temporada_alta <- promedio_reservas * 1.2  ## se usó 1.2 y 0.8
para poder definir lo que es una temporada baja y alta
```

```
temporada_baja <- promedio_reservas * 0.8  ## si es más del 20%
abajo del promedio o si es más del 20% arriba del promedio.
> temporada_baja
[1] 7959.333
> temporada_alta
[1] 11939

## se le da los valores de las temporadas a los meses
temporada <- cut(reservas_por_mes, breaks = c(-Inf,
temporada_baja, temporada_alta, Inf), labels = c("baja",
"media", "alta"))
##inf y -inf son necesarios en la funcion cut porque indican de
donde a donde van los valores de los promedios de las reservas

temporada (de todos los meses)
 [1] baja  media media media media media alta  alta  media
media baja  baja
Levels: baja media alta

# resumen final de los meses con su número de reservas totales
y la temporada a la que pertenecen
resumen_temporadas <- data.frame(Mes = names(reservas_por_mes),
Total_Reservas = as.numeric(reservas_por_mes), Temporada =
temporada)
print (resumen_temporadas)
```

	Mes	Total Reservas	Temporada
1	January	5929	baja
2	February	8068	media
3	March	9794	media
4	April	11089	media
5	May	11791	media
6	June	10939	media
7	July	12661	alta
8	August	13877	alta
9	September	10508	media
10	October	11160	media
11	November	6794	baja
12	December	6780	baja

Así se puede conocer (con los valores que hemos dado) cuales son las temporadas bajas, medias y altas

D. ¿Cuándo es menor la demanda de reservas?

```
## utilizando la pregunta anterior , filtramos solo con los
meses de temporada baja
meses_temporada_baja <-
resumen_temporadas[resumen_temporadas$Temporada == "baja", ]
print(meses_temporada_baja)
```

```
    Mes Total_Reservas Temporada
1   January           5929      baja
11 November           6794      baja
12 December           6780      baja
```

Con esta tabla se puede concluir que los meses donde de reservas es baja es en enero, noviembre, diciembre.

E. ¿Cuántas reservas incluyen niños y/o bebes?

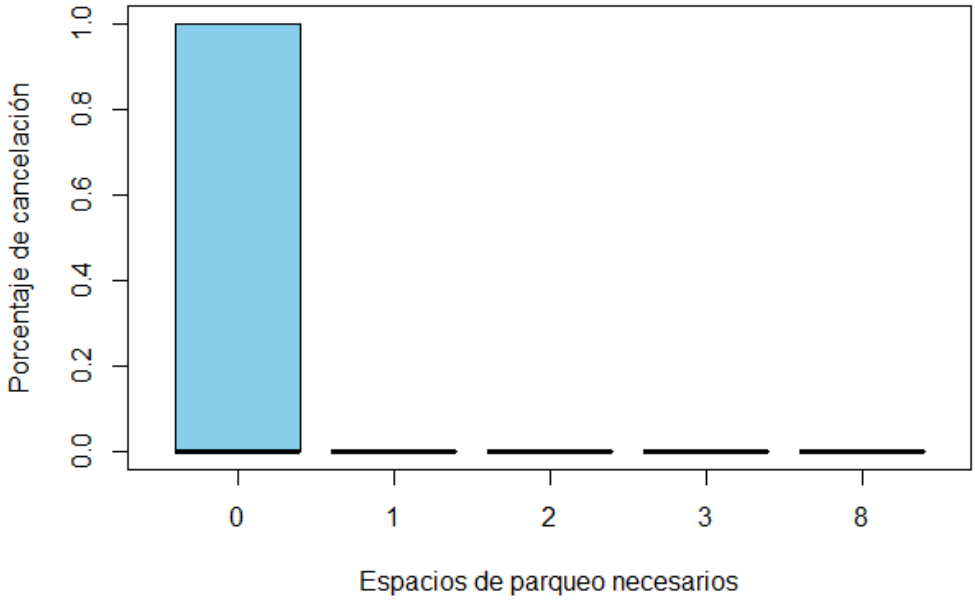
```
## filtramos las filas donde hayan niños o bebés
reservas_con_ninos_o_bebes <-
hotels_data.limpia[hotels_data.limpia$children > 0
|hotels_data.limpia$babies > 0, ]
##conteo
total_reservas_con_ninos_o_bebes <-
nrow(reservas_con_ninos_o_bebes)
print(total_reservas_con_ninos_o_bebes)
[1] 9336
```

Con este filtrado de los datos y conteo de estos mismo podemos determinar cuántas reservas incluyen niños.

F. ¿Es importante contar con espacios de estacionamiento?

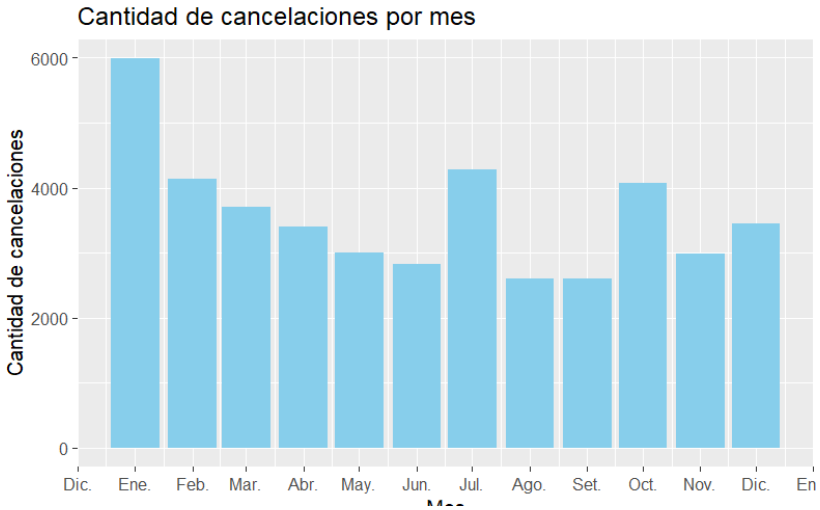
```
boxplot(hotels_data.limpia$is_canceled ~
hotels_data.limpia$required_car_parking_spaces, xlab="Espacios
de parque necesarios", ylab="Porcentaje de cancelación",
col="skyblue")
```

Este gráfico nos indica que los espacios de estacionamiento son importantes porque las cancelaciones de reservas se dan cuando los estacionamientos necesitados son 0



G. ¿En qué meses del año se producen más cancelaciones de reservas?

```
cancelaciones <- hotel %>% filter(reservation_status ==
"Canceled")
cancelaciones$reservation_status_date <-
as.Date(cancelaciones$reservation_status_date)
cancelaciones$mes <-
format(cancelaciones$reservation_status_date, "%m")
cancelaciones_por_mes <- cancelaciones %>% group_by(mes) %>%
summarise(cantidad = n())
cancelaciones_por_mes$mes <- as.Date(paste("2000",
cancelaciones_por_mes$mes, "01", sep = "-"))
ggplot(cancelaciones_por_mes, aes(x = mes, y = cantidad)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  scale_x_date(date_labels = "%b", date_breaks = "1 month") +
  labs(title = "Cantidad de cancelaciones por mes", x = "Mes",
y = "Cantidad de cancelaciones")
```



Como se puede ver en la gráfica, la cantidad de cancelaciones se realizan por el mes de enero, dichos datos se obtuvieron a traves de las variables reservation_status y reservation_status_date

III. Conjunto de datos

Los datos que tenemos son los siguientes:

```
'data.frame':   119390 obs. of  32 variables:
 $ hotel                         : chr  "Resort Hotel" "Resort Hotel" "Resort Hotel" "Resort Hotel" ...
 $ is_canceled                   : int  0 0 0 0 0 0 0 0 1 1 ...
 $ lead_time                     : int  342 737 7 13 14 14 0 9 85 75 ...
 $ arrival_date_year             : int  2015 2015 2015 2015 2015 2015 2015 2015 2015 2015 ...
 $ arrival_date_month            : Factor w/ 12 levels "January","February",..: 7 7 7 7 7 7 7 7 7 7 ...
 $ arrival_date_week_number      : int  27 27 27 27 27 27 27 27 27 27 ...
 $ arrival_date_day_of_month     : int  1 1 1 1 1 1 1 1 1 1 ...
 $ stays_in_weekend_nights       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ stays_in_week_nights          : int  0 0 1 1 2 2 2 2 3 3 ...
 $ adults                        : int  2 2 1 1 2 2 2 2 2 2 ...
 $ children                      : chr  "0" "0" "0" "0" ...
 $ babies                        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ meal                          : chr  "BB" "BB" "BB" "BB" ...
 $ country                       : chr  "PRT" "PRT" "GBR" "GBR" ...
 $ market_segment                : chr  "Direct" "Direct" "Direct" "Corporate" ...
 $ distribution_channel          : chr  "Direct" "Direct" "Direct" "Corporate" ...
 $ is_repeated_guest             : int  0 0 0 0 0 0 0 0 0 0 ...
 $ previous_cancellations        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ previous_bookings_not_canceled: int  0 0 0 0 0 0 0 0 0 0 ...
 $ reserved_room_type            : chr  "C" "C" "A" "A" ...
 $ assigned_room_type            : Factor w/ 12 levels "A","B","C","D",..: 3 3 3 1 1 1 3 3 1 4 ...
 $ booking_changes               : int  3 4 0 0 0 0 0 0 0 0 ...
 $ deposit_type                  : chr  "No Deposit" "No Deposit" "No Deposit" "No Deposit" ...
 $ agent                         : int  236 366 184 304 240 240 304 303 240 15 ...
 $ days_in_waiting_list          : int  0 0 0 0 0 0 0 0 0 0 ...
 $ customer_type                 : chr  "Transient" "Transient" "Transient" "Transient" ...
 $ adr                           : num  0 0 75 75 98 ...
 $ required_car_parking_spaces   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ total_of_special_requests     : int  0 0 0 0 1 1 0 1 1 0 ...
 $ reservation_status            : chr  "Check-Out" "Check-Out" "Check-Out" "Check-Out" ...
 $ reservation_status_date       : chr  "2015-07-01" "2015-07-01" "2015-07-02" "2015-07-02" ...
 $ personal                      : int  386 355 461 153 214 395 493 193 271 186 ...
```

Como se puede visualizar, constamos de 32 tipos de datos en este dataset, así es como nos aseguramos de obtener la cantidad de variables para realizar la tabla de descripción de las variables.

Nombre	Descripción
hotel	Tipo de Hotel
is_canceled	valor que indica si la reserva fue cancelada o no
lead_time	Número de días transcurridos entre la fecha de reserva con la fecha de llegada
arrival_date_year	Año de fecha de llegada
arrival_date_month	Mes de fecha llegada
arrival_date_week_number	Número de semana de la fecha llegada
arrival_date_day_of_month	Día del mes de la fecha de llegada
stays_in_weekend_nights	Número de noches que el huespede se quedó o reservó en el hotel para fin de semana
stays_in_week_nights	Número de noches que el huespede se quedó o reservó en el hotel entre semana
adults	Número de adultos
children	Número de niños
babies	Número de bebes
meal	Tipo de comida agendada

country	País de origen
market_segment	Designación del segmento de mercado
distribution_channel	Canal de distribución de cocina
is_repeated_guest	Valor que indica si el nombre de la reserva fue de alguien repetido (1 -> si, 0 -> no)
previous_cancellations	Número de reservas anteriores que fueron cancelados por el cliente antes de la reserva actual
previous_bookingsd_not_canceled	Número de reservas anteriores no cancelados por el cliente antes de la reserva actual
reserved_room_type	Código del tipo de habitación de reserva
assigned_room_type	Código del tipo de habitación asignado al reservar
booking_changes	Número de cambios realizadas a la reserva desde el momento en que la reserva se realizó hasta el momento de cancelación o entrega
deposit_type	Indicación si el cliente realizó un depósito para garantizar la reserva
agent	ID de la agencia de viajes que realizó la reserva
company	ID de la compañía que realizó la reserva o responsable del pago de la reserva
days_in_waiting_list	Número de días que la reserva estuvo en la lista de espera antes de confirmarse por el cliente
customer_type	Tipo de reserva

adr	Tarifa diaria promedio
required_car_parking_spaces	Número de plazas de aparcamiento necesarias para el cliente
total_of_special_request	Número de solicitudes especiales realizadas por el cliente
reservation_status	Último estado de la reserva
reservantion_status_date	Fecha en la que se cambió el último estado

IV. Análisis Exploratorio de Datos

Carga de datos:

Este código está escrito en R y se utiliza para leer un archivo CSV llamado "hotel_bookings.csv", que se encuentra en la ruta especificada por setwd(). Aquí está una explicación línea por línea:

```
## Cambiar la ruta en la que se encuentra el archivo a leer
setwd("C:/Users/ASUS/OneDrive/Escritorio/Quintociclo/Fundamento_Data_Science/Trabajo_Parcial")
```

Esta línea establece el directorio de trabajo (working directory) en R en la ruta especificada, donde se encuentra el archivo "hotel_bookings.csv".

```
hotels <- read.csv('hotel_bookings.csv', header = TRUE , sep = ',', dec = '.',stringsAsFactors = FALSE , na.strings = "")
## Estas líneas donde se crean los objetos hotels_data_limpia indican que se omitieron los valores NA y que se borró la columna "Company" porque no era de utilidad además eran NA'S
hotels_data.limpia <- na.omit(hotels)
hotels_data.limpia <- subset(hotels, select = -company)
```

Inspeccionar datos:

- Verificar tipos de datos de las columnas y nombres de las columnas:

```
column_info <- function(hotels_data.limpia) {
  cat("Nombres de las columnas:\n")
  print(names(hotels_data.limpia))
  cat("\nTipos de datos de las columnas:\n")
  print(sapply(hotels_data.limpia, class))
}

column_info(hotels_data.limpia )
```

```
Nombres de las columnas:
 [1] "hotel"                          "is_canceled"                    "lead_time"
 [4] "arrival_date_year"              "arrival_date_month"             "arrival_date_week_number"
 [7] "arrival_date_day_of_month"      "stays_in_weekend_nights"        "stays_in_week_nights"
[10] "adults"                         "children"                       "babies"
[13] "meal"                           "country"                        "market_segment"
[16] "distribution_channel"           "is_repeated_guest"              "previous_cancellations"
[19] "previous_bookings_not_canceled" "reserved_room_type"             "assigned_room_type"
[22] "booking_changes"                "deposit_type"                   "agent"
[25] "days_in_waiting_list"           "customer_type"                  "adr"
[28] "required_car_parking_spaces"    "total_of_special_requests"      "reservation_status"
[31] "reservation_status_date"        "personal"

Tipos de datos de las columnas:
                        hotel                   is_canceled                     lead_time
                  "character"                     "integer"                     "integer"
            arrival_date_year           arrival_date_month       arrival_date_week_number
                    "integer"                     "character"                     "integer"
     arrival_date_day_of_month         stays_in_weekend_nights           stays_in_week_nights
                    "integer"                     "integer"                     "integer"
                       adults                      children                        babies
                    "integer"                     "character"                     "integer"
                         meal                       country                  market_segment
                  "character"                     "character"                     "character"
          distribution_channel             is_repeated_guest          previous_cancellations
                  "character"                     "integer"                     "integer"
previous_bookings_not_canceled           reserved_room_type              assigned_room_type
                    "integer"                     "character"                     "character"
               booking_changes                    deposit_type                           agent
                    "integer"                     "character"                     "character"
          days_in_waiting_list                   customer_type                             adr
                    "integer"                     "character"                     "numeric"
   required_car_parking_spaces      total_of_special_requests            reservation_status
                    "integer"                     "integer"                     "character"
       reservation_status_date                      personal
                  "character"                     "integer"
```

La función “column_info” está diseñada para proporcionar información sobre la estructura de un dataframe en R. Al llamar a esta función con un dataframe como argumento, imprimirá los nombres de las columnas seguidos de los tipos de datos de cada columna. Esto te permite obtener una rápida visión general de la composición del dataframe, incluyendo qué variables están presentes y qué tipo de datos contienen. Es una herramienta útil para comenzar a explorar y comprender tus datos en R.

Procesar datos:

-Identificación de datos faltantes (NA)

```
      # Convertir la columna "agent" a tipo numérico (si es necesario)
hotels_data.limpia$agent <- as.numeric(as.character(hotels_data.limpia$agent))
      # Convertir la columna "agent" a enteros
hotels_data.limpia$agent <- as.integer(hotels_data.limpia$agent)
      # Verificar el cambio
Valor_Na <- function(x){
  sum=0
  for(i in 1 :ncol(x))
  {
    cat("en la columna",colnames(x[i]),"total de valores NA",colSums(is.na(x[i])),"\n")
  }
}
```

Valor_Na(hotels_data.limpia)

```
en la columna hotel total de valores NA 0
en la columna is_canceled total de valores NA 0
en la columna lead_time total de valores NA 0
en la columna arrival_date_year total de valores NA 0
en la columna arrival_date_month total de valores NA 0
en la columna arrival_date_week_number total de valores NA 0
en la columna arrival_date_day_of_month total de valores NA 0
en la columna stays_in_weekend_nights total de valores NA 0
en la columna stays_in_week_nights total de valores NA 0
en la columna adults total de valores NA 0
en la columna children total de valores NA 0
en la columna babies total de valores NA 0
en la columna meal total de valores NA 0
en la columna country total de valores NA 0
en la columna market_segment total de valores NA 0
en la columna distribution_channel total de valores NA 0
en la columna is_repeated_guest total de valores NA 0
en la columna previous_cancellations total de valores NA 0
en la columna previous_bookings_not_canceled total de valores NA 0
en la columna reserved_room_type total de valores NA 0
en la columna assigned_room_type total de valores NA 0
en la columna booking_changes total de valores NA 0
en la columna deposit_type total de valores NA 0
en la columna agent total de valores NA 16340
en la columna days_in_waiting_list total de valores NA 0
en la columna customer_type total de valores NA 0
en la columna adr total de valores NA 0
en la columna required_car_parking_spaces total de valores NA 0
en la columna total_of_special_requests total de valores NA 0
en la columna reservation_status total de valores NA 0
en la columna reservation_status_date total de valores NA 0
```

El código proporcionado realiza una serie de operaciones en la columna "agent" del dataframe hotels_data.limpia. Primero, verifica si hay datos faltantes en esta columna. Luego, convierte los valores de la columna a tipo numérico y posteriormente a enteros. Finalmente, verifica el cambio realizado. Esto garantiza que la columna "agent" esté lista para su análisis, asegurando consistencia en el tipo de datos y resolviendo cualquier problema de datos faltantes que pudiera existir.

-Explicación y aplicación de la técnica utilizada para eliminar o completar los datos faltantes

```
# Generar valores aleatorios enteros en el rango de 100 a 400
```

valores_enteros <- sample(100:400, sum(is.na(hotels_data.limpia$agent)), replace = TRUE)

```
# Reemplazar los NA en la columna "agent" con los valores aleatorios
generados
```

hotels_data.limpia$agent[is.na(hotels_data.limpia$agent)] <-

ifelse(is.na(hotels_data.limpia$agent), valores_enteros, hotels_data.limpia$agent)

str(hotels_data.limpia$agent)

head(hotels_data.limpia$agent)

```
> str(hotels_data.limpia$agent)
 int [1:119390] 282 383 113 304 240 240 304 303 240 15 ...
> head(hotels_data.limpia$agent)
[1] 282 383 113 304 240 240
```

El código presentado aborda la gestión de datos faltantes en la columna "agent" del dataframe hotels_data.limpia. Comienza generando valores aleatorios enteros en el rango de 100 a 400, cuya cantidad corresponde al número de valores faltantes en la columna "agent". Estos valores se asignan a las ubicaciones de los datos faltantes utilizando una indexación booleana. Posteriormente, se verifica que la columna "agent" ahora contenga valores enteros en lugar de NA, asegurando así que los datos estén completos y listos para su análisis ulterior. Este enfoque de imputación aleatoria proporciona una solución práctica cuando la pérdida de datos es aleatoria y no se puede inferir su valor fácilmente de otras variables.

-Identificación de datos atípicos (Outliers).

```
# Calcular la media y la desviación estándar de la columna 'adr'

mean_adr <- mean(hotels_data.limpia$adr)

sd_adr <- sd(hotels_data.limpia$adr)

# Definir límites para identificar valores atípicos (por ejemplo, valores más allá de 3 desviaciones estándar de la media)

limite_inferior <- mean_adr - 3 * sd_adr

limite_superior <- mean_adr + 3 * sd_adr

# Identificar valores atípicos

valores_atipicos <- hotels_data.limpia$adr[hotels_data.limpia$adr < limite_inferior | hotels_data.limpia$adr > limite_superior]
```

```
# Mostrar los valores atípicos
```

print(valores_atipicos)

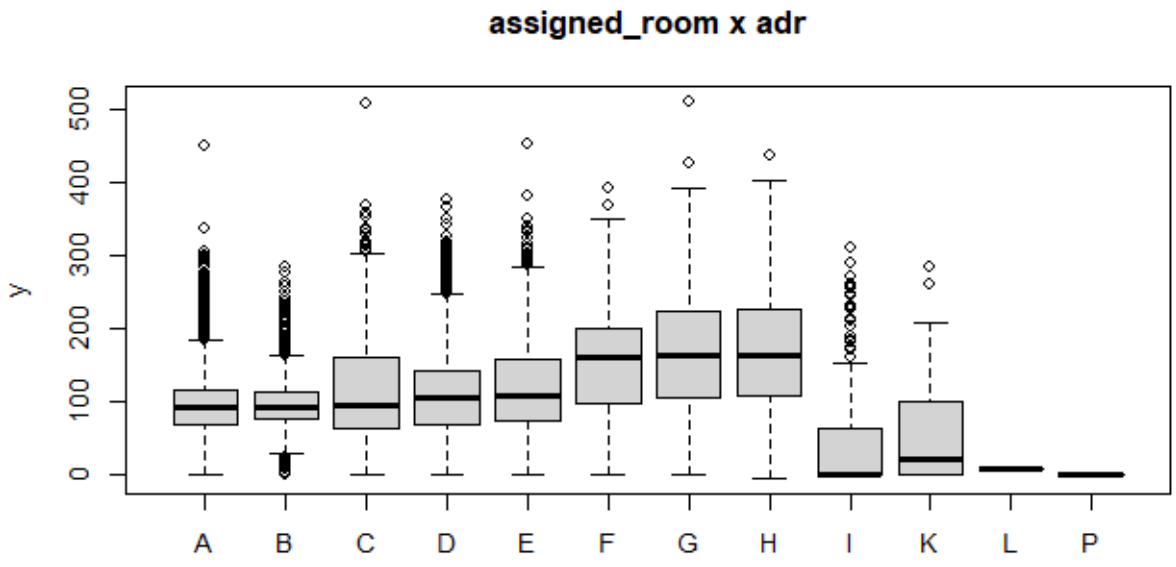
```
  [1]  280.74  268.00  267.00  277.50  276.43  277.00  254.00  274.93  258.33  255.00  266.40  271.00  266.00
 [14]  262.00  299.33  260.71  259.00  261.40  332.00  270.00  276.60  272.00  260.00  280.00  287.00  259.00
 [27]  259.00  288.00  262.00  259.00  292.00  259.00  266.50  253.57  259.00  256.50  282.00  283.32  272.70
 [40]  259.00  299.00  298.00  289.00  262.00  274.00  299.00  273.00  269.00  269.00  254.00  259.00  259.00
 [53]  369.00  262.00  278.60  271.00  254.31  261.50  259.00  256.00  259.00  291.00  279.00  259.00  254.00
 [66]  277.67  299.00  258.00  269.00  263.00  309.00  289.90  261.00  259.00  256.00  314.50  266.50  258.00
 [79]  286.79  281.00  274.00  275.00  256.75  288.00  259.00  274.00  304.00  286.00  329.00  281.00  274.00
 [92]  271.00  322.00  287.00  269.00  265.67  262.00  322.00  269.00  264.00  254.00  292.40  340.00  384.00
[105]  302.11  382.00  275.00  275.00  262.50  260.00  273.00  261.00  311.00  258.00  288.00  265.00  260.00
[118]  300.86  292.00  259.86  264.00  265.00  274.67  270.00  303.00  293.00  275.00  260.00  270.00  260.00
[131]  272.00  293.33  311.00  289.60  338.00  302.00  342.29  274.45  290.67  290.67  260.00  255.00  255.00
[144]  306.00  303.00  289.00  278.00  289.50  317.00  353.00  257.00  315.00  265.00  278.57  315.00  277.00
[157]  270.71  277.00  292.00  305.00  299.00  262.00  292.00  255.00  255.00  267.86  266.50  292.00  292.00
[170]  257.00  279.50  310.00  257.00  315.00  278.00  315.00  287.00  363.00  363.00  305.00  266.60  266.60
[183]  257.00  311.70  294.86  299.00  309.10  257.00  266.30  301.43  258.27  315.71  450.00  264.00  300.40
[196]  300.40  257.34  266.00  282.29  270.00  269.50  283.60  378.00  358.75  328.00  330.00  259.33  292.00
[209]  269.00  298.00  378.00  310.00  262.00  323.00  297.00  269.00  259.00  303.70  294.50  275.00  378.00
[222]  317.00  262.00  298.00  262.00  292.00  292.00  297.00  259.00  258.00  392.00  294.50  280.00  273.25
[235]  270.00  264.29  259.43  262.00  297.38  300.00  303.20  340.00  256.57  262.00  261.00  437.00  264.00
[248]  388.00  274.50  262.00  254.00  310.00  262.00  297.00  308.00  259.00  273.00  270.00  310.00  264.00
[261]  308.00  262.00  289.80  378.00  338.00  311.50  318.00  289.80  290.00  260.00  270.00  290.00  270.00
[274]  282.00  262.00  270.00  330.00  265.00  257.60  257.60  340.00  270.00  268.00  343.00  254.00  289.60
[287]  255.45  255.45  260.00  270.00  340.00  270.00  297.00  378.00  295.50  278.14  260.00  280.00  319.00
[300]  268.00  270.00  255.00  260.00  270.00  331.33  259.00  303.33  295.67  295.67  342.17  262.38  284.86
[313]  340.86  299.43  294.29  293.86  290.00  290.00  295.00  289.80  253.80  286.25  341.00  312.00  297.00
[326]  265.00  293.60  328.00  284.00  290.00  254.00  305.00  335.00  288.10  275.25  367.00  508.00  317.00
[339]  318.82  256.00  297.50  266.00  279.00  271.00  274.00  258.43  264.50  256.50  259.00  284.00  300.00
[352]  305.00  305.00  305.00  305.00  274.00  254.00  282.00  311.33  263.57  262.50  287.50  271.00  271.00
[365]  296.00  271.00  295.00  257.00  276.60  271.00  259.00  278.00  281.00  318.00  276.00  259.00  308.40
[378]  274.00  301.00  260.33  281.00  302.86  288.10  257.00  307.00  264.00  274.00  282.00  265.83  263.00
[391]  340.71  283.00  283.20  273.50  263.00  336.50  259.00  318.71  268.33  352.00  335.00  359.00  253.50
[404]  310.00  353.67  269.00  269.00  276.00  313.71  271.00  262.60  288.00  268.00  269.00  276.00  256.00
```

El código proporcionado se centra en identificar valores atípicos dentro de la columna 'adr', que representa el precio promedio diario en el dataframe 'hotels_data.limpia'. Comienza calculando la media y la desviación estándar de los precios diarios para establecer límites que definan qué valores se considerarán atípicos. Aquellos precios que se desvíen más allá de tres veces la desviación estándar de la media son etiquetados como atípicos. Estos valores se recopilan y se muestran, lo que permite una identificación clara de los datos que podrían requerir una atención especial durante el análisis subsiguiente. Este proceso es esencial para comprender la distribución de los datos y garantizar la integridad de los resultados analíticos al considerar la posible influencia de valores extremos.

- Explicación y aplicación de la(s) técnica(s) utilizada(s) para transformar los datos atípicos.

```
# La columna 'assigned_room_type' contiene los tipos de habitación
asignados (por ejemplo, 'A', 'B', 'C', etc.).

# La columna 'adr' representa el precio promedio diario.
```

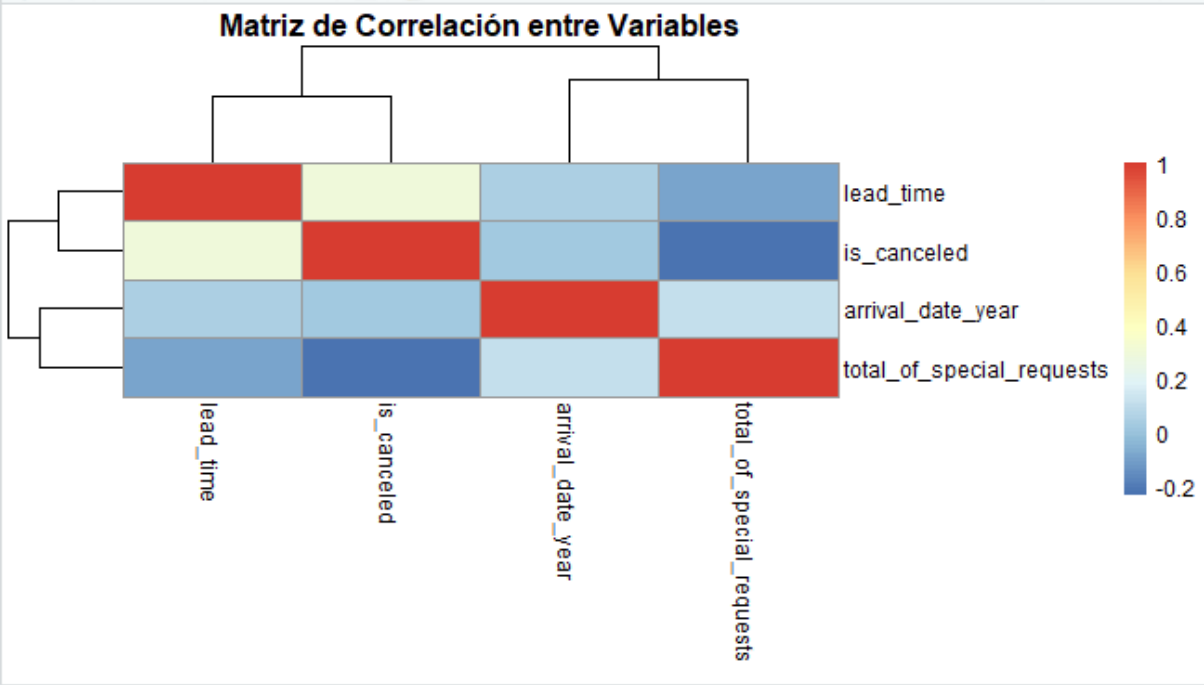
```
# Paso 1: Conversión de la columna 'assigned_room_type' a factor

hotels_data.limpia$assigned_room_type <-
as.factor(hotels_data.limpia$assigned_room_type)

# Paso 2: Creación de un gráfico de dispersión original

plot(hotels_data.limpia$assigned_room_type, hotels_data.limpia$adr)

# Este gráfico muestra la relación entre los tipos de habitación asignados
y los precios promedio diarios.

# Paso 3: Filtrado de datos para valores de 'adr' menores a 5000

HB2 <- hotels_data.limpia %>% filter(hotels_data.limpia$adr < 5000)

# Paso 4: Creación de un segundo gráfico de dispersión con datos filtrados

plot(HB2$assigned_room_type, HB2$adr, main = "assigned_room x adr")

# Este gráfico muestra la relación entre los tipos de habitación asignados
y los precios promedio diarios,

# pero solo para aquellos registros con precios menores a 5000.
```



En el código proporcionado, se trabaja con un conjunto de datos llamado ‘hoteles’. Primero, se convierte la columna ‘assigned_room_type’ en un factor para representar los diferentes tipos de habitaciones asignadas a los huéspedes en un hotel. Luego, se crea un gráfico de dispersión para visualizar la relación entre estos tipos de habitaciones y el precio promedio diario (adr). Posteriormente, se filtran los datos para incluir sólo aquellos registros con precios de habitación menores a 5000. Finalmente, se genera un segundo gráfico de dispersión utilizando los datos filtrados, enfocándose en los precios más bajos. El objetivo es explorar cómo los diferentes tipos de habitaciones afectan los precios promedio en el contexto hotelero.

- **Visualizar datos:**

```
# Selecciona las variables relevantes (puedes agregar más según tus necesidades)

selected_vars <- hotels_data.limpia[, c('lead_time', 'is_canceled', 'arrival_date_year', 'total_of_special_requests')]

# Calcula la matriz de correlación

cor_matrix <- cor(selected_vars)

# Instala y carga el paquete 'pheatmap' si aún no lo has hecho

# install.packages('pheatmap')

library(pheatmap)

# Crea un mapa de calor para visualizar las correlaciones

pheatmap(cor_matrix, scale = 'none', main = "Matriz de Correlación entre Variables")
```



El código proporcionado se utiliza para analizar las correlaciones entre variables seleccionadas en un conjunto de datos. Al seleccionar cuidadosamente las variables relevantes para el análisis, se calcula una matriz de correlación y se representa visualmente mediante un mapa de calor. Esta visualización permite identificar patrones de asociación y comprender mejor cómo interactúan las diferentes características del conjunto de datos. Este enfoque analítico proporciona una base sólida para inferir conclusiones sobre las relaciones entre las variables, respaldando así la toma de decisiones informadas y el avance en la comprensión del problema investigado. En última instancia, este proceso de análisis de correlación contribuye significativamente a la generación de conocimiento en el campo de estudio relevante.

V. Conclusiones

- La limpieza de dataset es un paso importante para poder recuperar valores que están encubiertos por un “NA”, también se puede usar el método de eliminación de columnas ya que no son datos relevantes para el análisis del dataset.

- Se observa que la gente prefiere el tipo "City Hotel" en comparación con "Resort Hotel". Esto se evidencia por el mayor número de reservas realizadas en City Hotel en comparación con Resort Hotel.

- Después de un aumento en la demanda hasta 2016, se observa una tendencia a la baja en las estancias en fines de semana a partir de entonces. Esto puede indicar un cambio en las preferencias de los clientes o cambios en el mercado que podrían afectar la demanda.

- Se identifican tres temporadas: baja, media y alta. Los meses de temporada baja son enero, noviembre y diciembre. La temporada alta abarca julio y agosto, mientras que el resto de los meses se consideran de temporada media.

- Un total de 9336 reservas incluyen niños o bebés. Esto sugiere que hay una demanda considerable de alojamientos que puedan acomodar a familias.

- Se observa que las cancelaciones de reservas son más frecuentes cuando no hay espacios de estacionamiento disponibles. Esto indica que contar con espacios de estacionamiento puede influir significativamente en la decisión de reserva de los clientes.

- Para futuros análisis, una recomendación precisa es usar la función pasar variables a factor para se puedan hacer reportes o resúmenes de las tablas del dataset, por el contrario lo leería como un “character” y por ende no te daría los datos que buscas de forma ordenada.

- Como lección nos queda que todo reporte de datos, nos sirve poder gestionar de una manera correcta y precisa el análisis de los futuros datasets a utilizar. Además la creación de una matriz de correlación de variables, es de vital importancia ya que sirve para entender la naturaleza y la fuerza de las relaciones lineales entre variables en un conjunto de datos.

VI. Bibliografía

- António, N., Almeida, A., & Nunes, L. (2019). *Hotel booking demand datasets. Data in Brief, 22, 41–49*. Recuperado de: https://doi.org/10.1016/j.dib.2018.11.126

- Paradinas, I. (2021). *Curso R base*. Recuperado de: https://bookdown.org/paradinas_iosu/CursoR/

VII. Anexos

Link del GitHub: Nemesisian/CC216--TP-2024-1- (github.com)